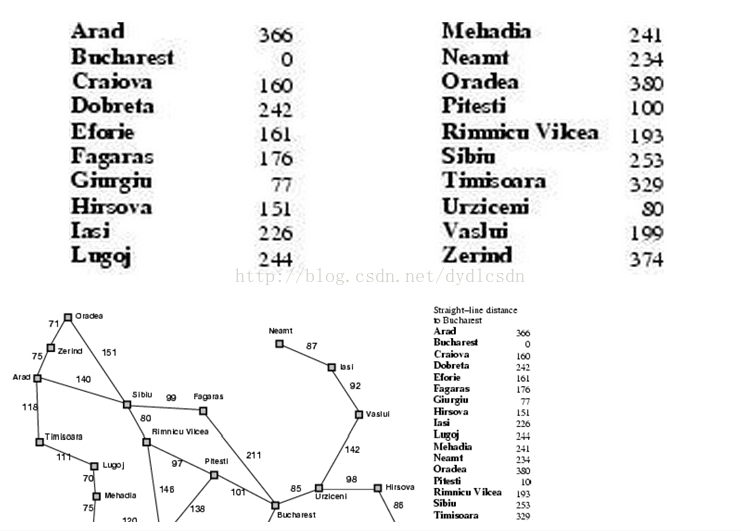
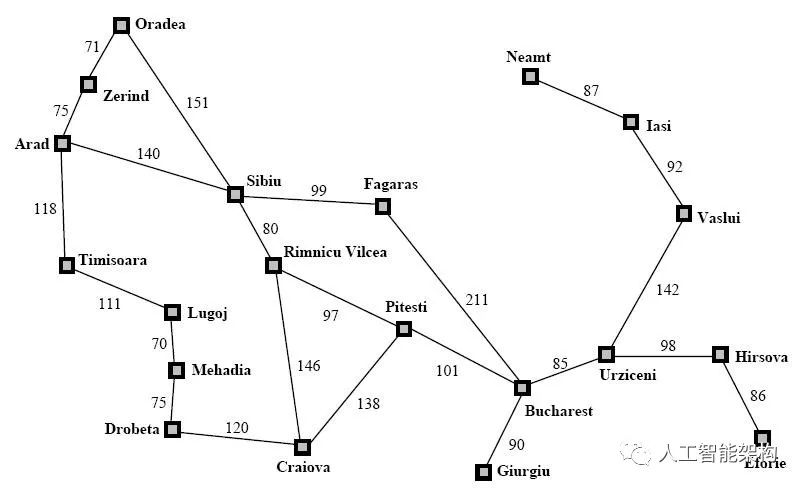
# A\* 算法解决罗马尼亚问题

## 问题的提出：

**通过使用A\* 得到从起始点Arad到目标点Bucharest的一条路径，即为罗马尼亚问题的一个解，在求解的过程中记录每种算法得到的解，即输出每种解得到的条路径。**

## 罗马尼亚地图：

****

****

## 实验内容

**import** java.util.Stack;

**public** **class** Ass {

**int** result;

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

node start=**new** node(0);

node end=**new** node(12);

System.***out***.println("----------------罗马尼亚问题---------------");

*ass*(start,end);

}

**public** **static** **class** Graph{

**public** **int** path[][]=**new** **int**[][]{

{0,75,10000,118,140,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000},

{75,0,71,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000},

{10000,71,0,10000,151,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000},

{118,10000,10000,0,10000,111,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000},

{140,10000,151,10000,0,10000,80,99,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000},

{10000,10000,10000,111,10000,0,10000,10000,70,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000},

{10000,10000,10000,10000,80,10000,0,10000,10000,10000,146,97,10000,10000,10000,10000,10000,10000,10000,10000},

{10000,10000,10000,10000,99,10000,10000,0,10000,10000,10000,10000,211,10000,10000,10000,10000,10000,10000,10000},

{10000,10000,10000,10000,10000,70,10000,10000,0,75,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000},

{10000,10000,10000,10000,10000,10000,10000,10000,75,0,120,10000,10000,10000,10000,10000,10000,10000,10000,10000},

{10000,10000,10000,10000,10000,10000,146,10000,10000,120,0,138,10000,10000,10000,10000,10000,10000,10000,10000},

{10000,10000,10000,10000,10000,10000,97,10000,10000,10000,138,0,101,10000,10000,10000,10000,10000,10000,10000},

{10000,10000,10000,10000,10000,10000,10000,211,10000,10000,10000,101,0,90,85,10000,10000,10000,10000,10000},

{10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,90,0,10000,10000,10000,10000,10000,10000},

{10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,85,10000,0,98,10000,142,10000,10000},

{10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,98,0,86,10000,10000,10000},

{10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,86,0,10000,10000,10000},

{10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,142,10000,10000,0,92,10000},

{10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,92,0,87},

{10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,87,0}

};

**public** **int**[]h=**new** **int**[]{366,374,380,329,253,244,193,176,241,242,160,100,0,77,80,151,161,199,226,234};

**public** String[] cities=**new** String[]{

"Arad","Zerind","Oradea","Timisoara",

"Sibiu","Lugoj","Rimnicu Vilcea",

"Fagaras","Mehadia","Dobreta","Craiova",

"Pitesti","Bucharest","Giurgiu","Urziceni",

"Hirsova","Eforie","Vaslui","Lasi","Neamt"

};

}

**public** **static** **class** node{

**int** num;

**int**[] suc=**new** **int**[20];

Graph a=**new** Graph();

node(**int** num)

{

**this**.num=num;

**for**(**int** i =0; i < 20; i++) {

**if**(a.path[num][i]>0&&a.path[num][i]<10000)

suc[i]=1;

**else**

suc[i]=0;

}

}

}

**public** **static** **void** ass(node start,node end) {

**int** length[]=**new** **int** [20];

**int** min;

**int** k=0;

**int** sum=0;

Stack<Integer> stack=**new** Stack<Integer>();

stack.push(start.num);

node next=start;

**while**(next.num!=end.num)

{

**int** p=next.num;

**for**(**int** i=0;i<20;i++) {

**if**(next.suc[i]!=0)

length[i]=next.a.path[next.num][i];

**else**

length[i]=10000;

}

**for**(**int** i=0;i<20;i++)

{

**if**(length[i]<1000) {

length[i]=length[i]+next.a.h[i];

}

}

min=length[0];

**for**(**int** i=0;i<20;i++)

{

**if**(length[i]<min) {

min=length[i];

next.num=i;

}

}

//System.out.print(length[next.num]+" ");

stack.push(next.num);

node aaa=**new** node(next.num);

next=aaa;

next.suc[p]=0;

k++;

}

**for**(**int** i=0;i<=k;i++) {

**if**(i<=k-1) {

System.***out***.println(stack.get(i)+" "+next.a.cities[stack.get(i)]+" "+next.a.path[stack.get(i)][stack.get(i+1)]);

sum=sum+next.a.path[stack.get(i)][stack.get(i+1)];

}

**else**

{

System.***out***.println(stack.get(i)+" "+next.a.cities[stack.get(i)]);

System.***out***.println("sum= "+sum);

}

}

}

}

## 实验结果：

